

Vial Counting Using Image Processing



By : M JEEVAN
Data Scientist



Project Leadership



Sharat Manikonda
Director at Innodatatics and Sponsor
[linkedin.com/in/sharat-chandra](https://www.linkedin.com/in/sharat-chandra)

Team Members

Name: M Jeevan

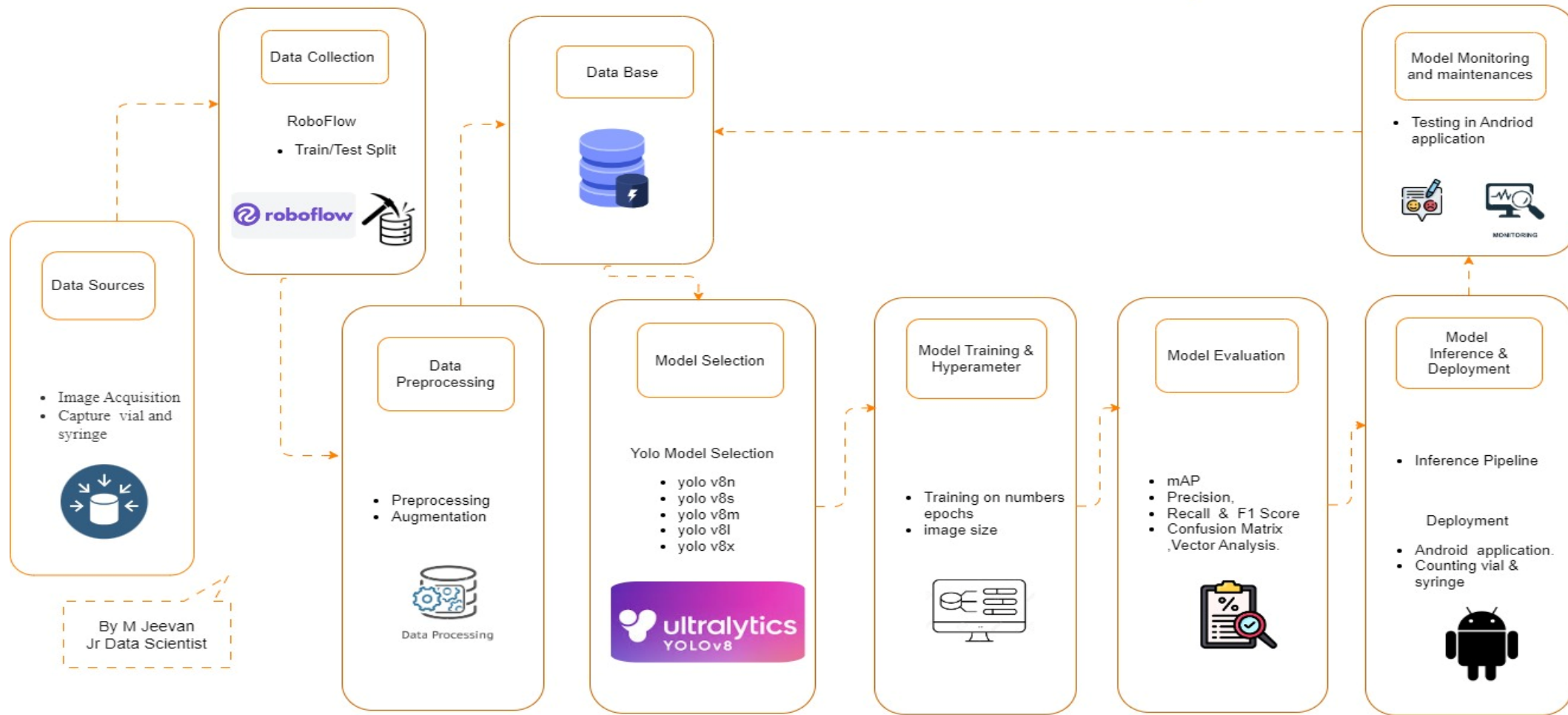
www.linkedin.com/in/m-jee8van78753

Contents

- Business objective
- Business Constraints
- Project Architecture
- Data collection and details
- Exploratory Data Analysis
- Visualization
- Modeling
- Evaluation
- Deployment

Project Overview and Scope

Vial Counting Using Image Processing by M Jeevan



Business Problem

- Client: One of the leading pharmaceutical company in India.
- The challenge is to design a counting system for syringes and vials that streamlines operations and mitigates the risk of human error, ultimately optimizing productivity and accuracy in the process.

Business Objective

Objective

Maximize efficiency and Minimize manual errors in inventory management.

Constraints

- Minimize the cost of solution.

CRISP-ML(Q) Methodology

There are six stages of CRISP-ML(Q) Methodology

1.Business and data understanding

2.Data preparation

3.model building

4.Model evaluation

5.Model deployment

6.Monitoring and maintenance

Technical Stacks

1. Machine Learning Framework:

- Ultralytics YOLOv8 (n, s, m, l, x)

2. Programming Languages:

- Python

3. Data Annotation:

- Manual annotation tools for labeling data (vial and syringe classes)

4. Libraries and Tools:

- OpenCV for image processing
- TensorFlow Lite for model inference and deployment
- NumPy and Pandas for data manipulation
- Albumentations for data augmentation

5. Model Training:

- Custom training with YOLOv8 for 100 epochs
- Model saving as `best.pt`

6. Model Inference and Deployment:

- Convert trained model to TensorFlow Lite format (`model.Tflite`)
- Deploy model in Android application

7. Data Management:

- Train/Test split for dataset preparation
- Data processing steps (normalization, augmentation, etc.)
- Dataset versioning and export in YOLOv8 format

8. Hardware Setup:

- Custom dome light camera setup for image capturing

9. Development Environment:

- Jupyter Notebooks for experimentation and development
- Google Colab or local machine with GPU for model training

10. Version Control:

- Git for source code management and collaboration

11. Deployment Platform:

- Android for mobile application deployment

12. Testing and Validation:

- Testing the model with vial and syringe camera setup
- Application testing on Android device

Data Collection and Understanding

1. Project Research:

- Understand problem domain and requirements.
- Identify key features of vials and syringes.

2. Build Flow Architecture:

- Design data flow and processing pipeline.

3. Data Collection:

- Gather diverse images of vials and syringes.

4. Manual Annotation:

- Label images with vial and syringe classes.
- Use annotation tools for bounding boxes,

5. Train/Test Split:

- Divide data into training and testing sets.
- Ensure balanced class distribution.

6. Data Processing:

- Normalize images to standard sizes/formats.
- Enhance image quality and consistency.

7. Data Augmentation:

- Apply techniques like rotation, scaling, and color adjustments.
- Increase dataset diversity.

8. Generate Dataset Versions:

- Create multiple dataset versions.
- Export in YOLOv8 format.

System Requirements

1. Software Requirements:

- **Operating System:** Windows 10/11
- **Python:** Python 3.11, packages (pandas, matplotlib, seaborn, scikit-learn, etc.)
- **Database:** MySQL
- **Data Processing :** Roboflow

2. Collaboration Tools:

- Google Meet for communication and coordination within our project team.

3. Backup and Recovery:

- Back up project data and code to prevent data loss using cloud storage.

Exploratory Data Analysis [EDA]

1. Data Inspection:
 - Load and preview dataset
 - Visualize sample images
2. Summary Statistics:
 - Analyze class distribution
 - Check image dimensions
3. Data Quality Checks:
 - Identify missing/corrupted data
 - Verify annotation accuracy
4. Data Visualization:
 - Plot images with annotations
 - Visualize class distribution
5. Statistical Analysis:
 - Bounding box size analysis
 - Correlation checks
6. Data Augmentation Analysis:
 - Evaluate augmentation techniques
 - Visualize augmented data
7. Train/Test Split Analysis:
 - Ensure balanced distribution
 - Inspect train/test samples
8. Save Processed Data:
 - Export cleaned data
 - Document EDA findings

Data Preprocessing

Preprocess Images

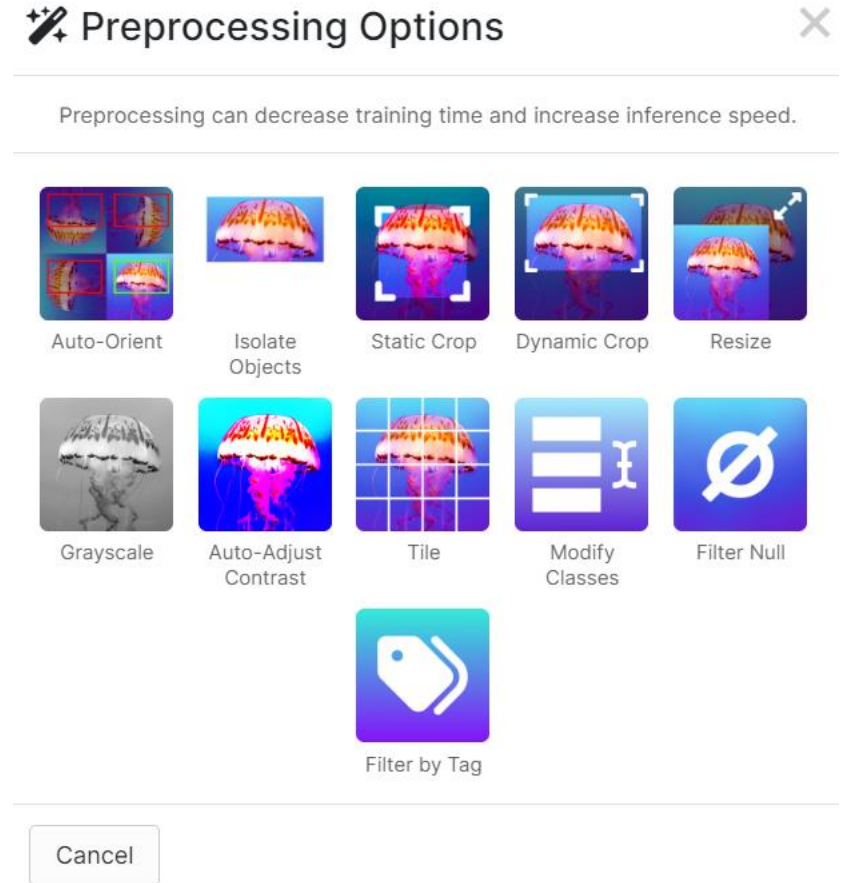
Image preprocessing steps to prepare data for models.

Preprocessing ensures your dataset is in a standard format (e.g. all images are the same size). This step is essential to ensure your dataset is consistent before training a model.

Preprocessing applies to all images in your Train, Valid, and Test set (unlike [Augmentations](#), which only apply to the Train set).

The Roboflow platform offers the following preprocessing options:

- Auto-Orient
- Resize
- Grayscale
- Auto-Adjust Contrast
- Isolate Objects
- Static Crop
- Tile
- Modify Classes
- Filter Null
- Filter by Tag



Data Augmentations


Create Augmented Images

Create augmented images to improve model performance.

Image augmentation is a step where augmentations are applied to existing images in your dataset. This process can help improve the ability of your model to generalize and thus perform more effectively on unseen images.






Roboflow supports the following augmentations:






- Flip
- 90 degree rotation
- Random rotation
- Random crop
- Random shear
- Blur
- Exposure
- Random noise
- Cutout (paid plans only)
- Mosaic (paid plans only)




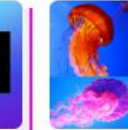
 Augmentation Options ✕

Augmentations create new training examples for your model to learn from.






IMAGE LEVEL AUGMENTATIONS



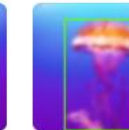

    
Flip 90° Rotate Crop Rotation Shear

    
Grayscale Hue Saturation Brightness Exposure

   
Blur Noise Cutout Mosaic

BOUNDING BOX LEVEL AUGMENTATIONS ?

    
Flip 90° Rotate Crop Rotation Shear

   
Brightness Exposure Blur Noise

Cancel

Model Building

YOLOv8 Performance: Benchmarked on COCO

Model	size (pixels)	mAP ^{val} 50-95	Speed CPU (ms)	Speed T4 GPU (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	-	-	3.2	8.7
YOLOv8s	640	44.9	-	-	11.2	28.6
YOLOv8m	640	50.2	-	-	25.9	78.9
YOLOv8l	640	52.9	-	-	43.7	165.2
YOLOv8x	640	53.9	-	-	68.2	257.8

YOLOv8 scored a 80.2% mAP score on Roboflow 100, compared to 73.5% mean score on YOLOv5. This shows that YOLOv8 is significantly better at domain-specific tasks than Ultralytics' YOLOv5 predecessor. We compared YOLOv5s and YOLOv8 in this analysis.

Model Accuracy Comparison

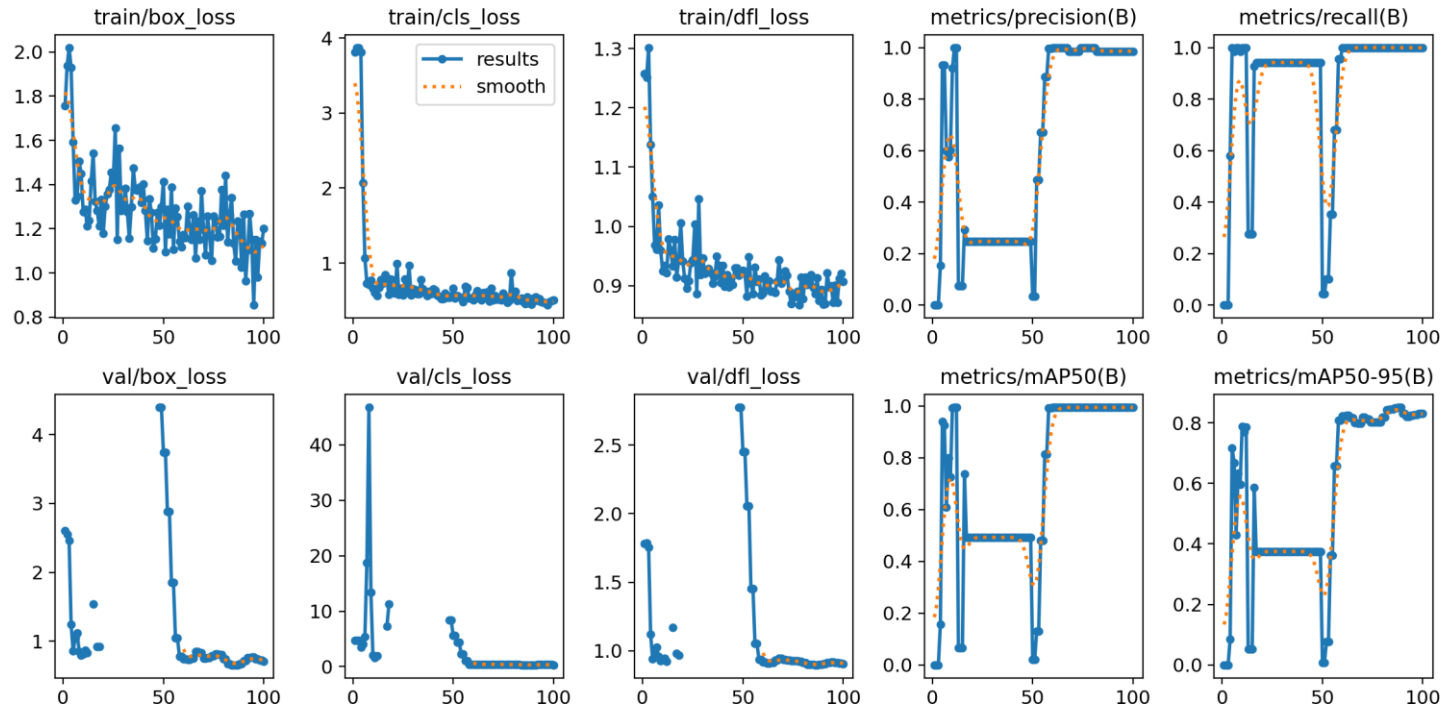
```
/content
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.3.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 268 layers, 68125494 parameters, 0 gradients, 257.4 GFLOPs
val: Scanning /content/datasets/Vial-Counting--15/valid/labels.cache... 2 images, 0 backgrounds, 0 corrupt: 100% 2/2 [00:00<?, ?it/s]
      Class      Images  Instances  Box(P      R      mAP50  mAP50-95):  0% 0/1 [00:00<?, ?it/s]WARNING ⚠ NMS time limit 0.600s exceeded
      Class      Images  Instances  Box(P      R      mAP50  mAP50-95): 100% 1/1 [00:01<00:00, 1.43s/it]
      all           2         161      0.772      0.444      0.445      0.359
      Vials          2           9      0.544      0.889      0.89       0.719
      syringe        2        152       1         0         0         0
Speed: 0.4ms preprocess, 120.1ms inference, 0.0ms loss, 502.9ms postprocess per image
Results saved to runs/detect/val
💡 Learn more at https://docs.ultralytics.com/modes/val
```

FIG: Yolo v8 m Accuracy

```
... /content
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.3.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 268 layers, 43608150 parameters, 0 gradients, 164.8 GFLOPs
val: Scanning /content/datasets/Vial-counting-2-4/valid/labels.cache... 1 images, 0 backgrounds, 0 corrupt: 100% 1/1 [00:00<?, ?it/s]
      Class      Images  Instances  Box(P      R      mAP50  mAP50-95):  0% 0/1 [00:00<?, ?it/s]WARNING ⚠ NMS time limit 0.550s exceeded
      Class      Images  Instances  Box(P      R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00, 1.02it/s]
      all           1         68      0.957      1         0.995      0.818
      vial           1         68      0.957      1         0.995      0.818
Speed: 0.4ms preprocess, 79.9ms inference, 0.0ms loss, 718.4ms postprocess per image
Results saved to runs/detect/val
💡 Learn more at https://docs.ultralytics.com/modes/val
```

FIG: Yolo v8 L Accuracy

Best Model – Yolo V8_L Model



- Precision:** Get accurate metrics like mAP50, mAP75, and mAP50-95 to comprehensively evaluate your model.

- Convenience:** Utilize built-in features that remember training settings, simplifying the validation process.

- Flexibility:** Validate your model with the same or different datasets and image sizes.

- Hyperparameter Tuning:** Use validation metrics to fine-tune your model for better performance.

Model Deployment - Strategy

Model Deployment Overview:

Highlight the two deployment strategies: Android app using .tflite and Streamlit using best.pt.
Emphasize the importance of model optimization and efficient integration.

Technical Implementation:

Detail the process of converting and optimizing models for each platform.
Explain the integration steps for TensorFlow Lite in Android and PyTorch in Streamlit.

User Experience:

Showcase the designed user interfaces for both the mobile app and web application.
Discuss the real-time processing capabilities and user-friendly features implemented.

Testing and Scalability:

Describe the testing procedures for ensuring compatibility and performance.
Outline the deployment strategies for both platforms and the measures taken to ensure scalability and stability.

Screen shot of output

Deploy

Object Detection with YOLOv8

Choose an image...

Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

Browse files

IMG_20240618_165324.jpg 3.0MB

Select class

syringe

Select confidence score

0.00

0.50

1.00

Uploaded image

Detect Objects

Processed image

Number of syringes detected: 290

Deploy

Object Detection with YOLOv8

Choose an image...

Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

Browse files

IMG_20240618_162639.jpg 4.1MB

Select class

vial

Select confidence score

0.00

0.50

1.00

Uploaded image

Detect Objects

Processed image

Number of vials detected: 125

Deploy

Object Detection with YOLOv8

Choose an image...

Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

Browse files

IMG_20240618_164358.jpg 4.0MB

Select class

vial

Select confidence score

0.00

0.50

1.00

Uploaded image

Detect Objects

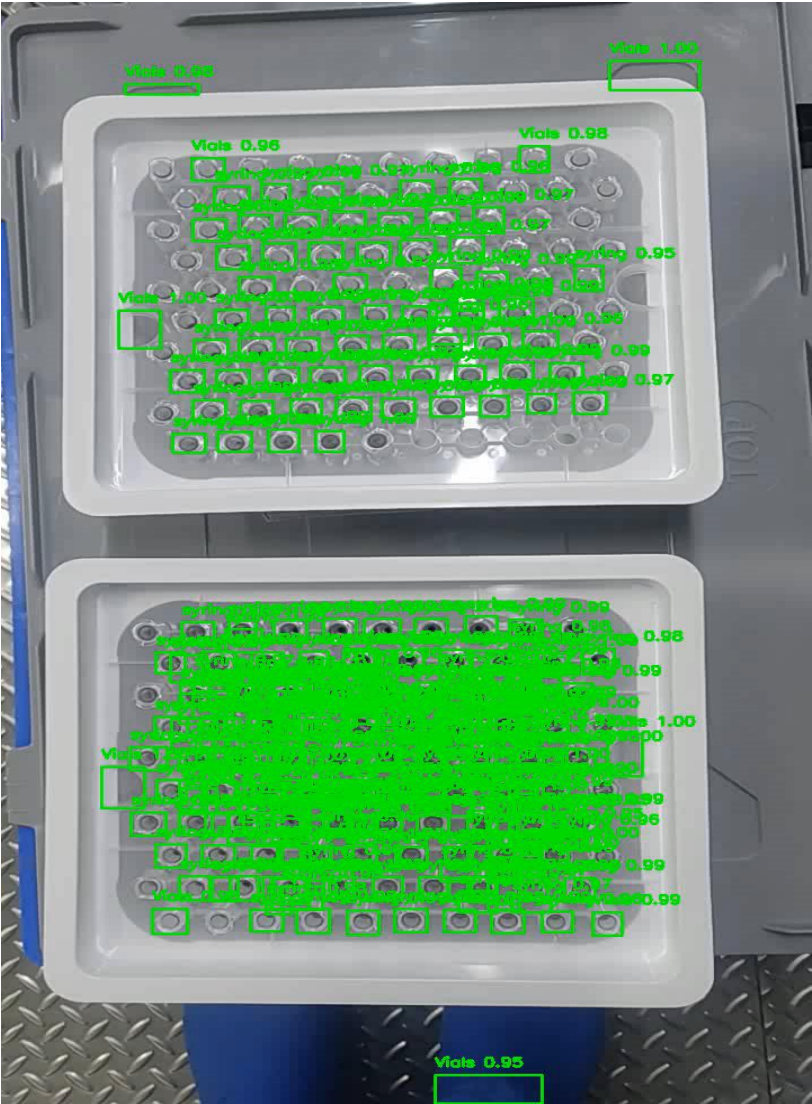
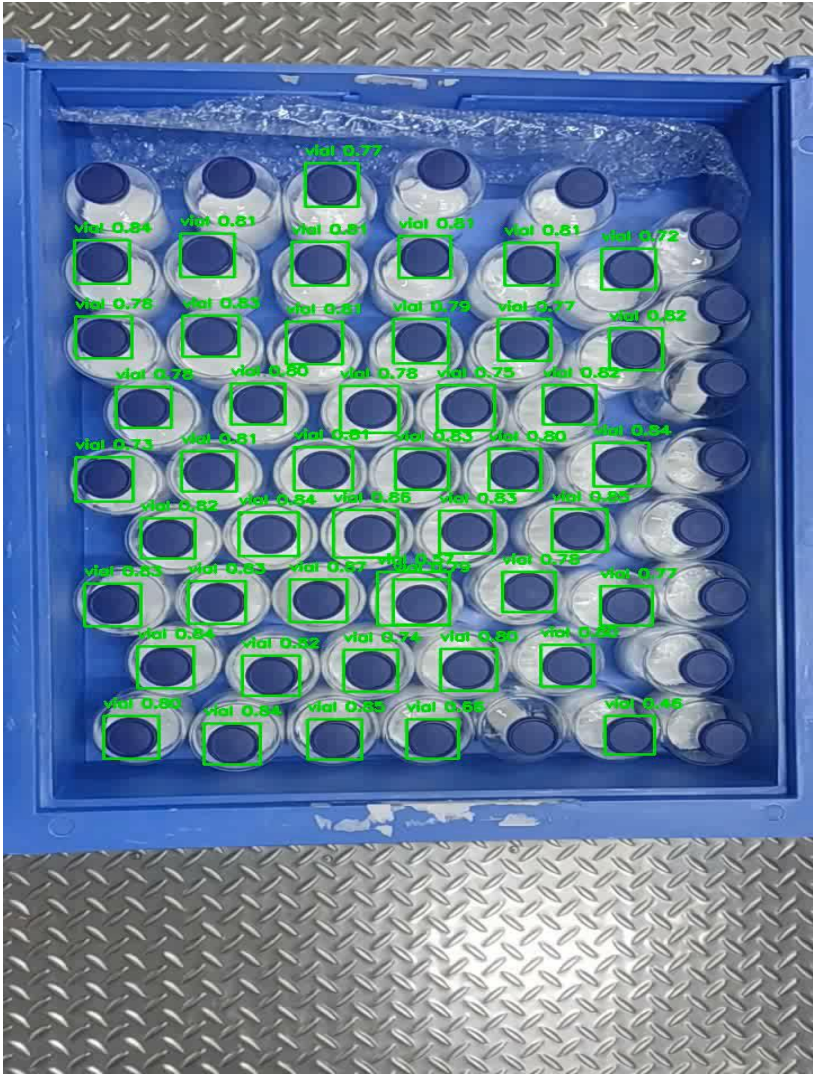
Processed image

Number of vials detected: 58

innodatatics

Innovation • Data • Analytics

Video of output



Challenges

Variability in Lighting Conditions:

- Ensuring consistent and optimal lighting with the custom dome setup to maintain image quality

Dataset Annotation Complexity:

- Accurately annotating vials and syringes with precise bounding boxes across a large dataset.

Real-Time Inference Requirements:

- Achieving fast and accurate model inference on Android devices without compromising performance.

Integration with Existing Systems:

- Ensuring seamless integration with current inventory management systems and data flow.

Regulatory Compliance and Certification:

- Meeting pharmaceutical industry standards and obtaining necessary certifications for deployment.

Future Scopes

Enhanced Model Capabilities:

Explore advanced object detection models beyond YOLOv8 for improved accuracy.
Incorporate deep learning techniques like transfer learning for faster convergence.

Integration with IoT Devices:

Implement real-time data streaming from IoT devices for continuous monitoring.
Enable remote management and alerts for inventory status.

Expansion to Other Pharmaceutical Products:

- Extend the system's capability to count and manage other pharmaceutical products.
- Adapt the solution for broader applications in healthcare and beyond.

Queries ?



